```
funct ( list <int > queries)                    — O(Q) × O(√N)
{
      for ( i = 0  →  queries . size)
      {
            list =  get Prime Fact  ( queries [i] );
            print ( list ) ;
      }
}
```

<-Naive approach

An optimal solution would be storing the smallest prime factor(SPF) for number till max(query).

The code SPF is similar to calculating is prime but here we replace all multiples of i (prime numbers) with i assuming number and index do not match.

Once, we have pre-computed SPF array we divide each query by it's SPF and add it to list of its sorted prime factors.

```
spf [10^5 + 1]
for ( i = 1  → 10^5 ) spf [i] = i          step 1

for ( i = 2 ; i x i <= 10^5 ; i++)
{
      if ( spf [i] == i)
      {
            for ( j = i x i ; j <= 10^5 ; j = j+i )
            {
                  if ( spf [j] == j)
                        spf [j] = i ;
            }
      }
}
```

This takes

N log(log N)

```
for ( i = 0  →  queries . size)
{
      n =  queries [i] ;                    O(
      while ( n != 1)                       step 2
      {
            print ( spf [n] ) ;
            n = n / spf [n] ;
      }
}
```

Q x log N

T.C -> N + log (log N) + (Q . $\log_2$ N)

SC -> O(N)